Weighted Finite Automata and Representation of Images

by

Kamala Krithivasan

Theoretical Computer Science Lab
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Acknowledgments

This work was done in collaboration with

S. V. Ramasubramanian

A. Siddartha Reddy and

Y. Sivasubramanyam
Addressing of Pixels

- A digital image of resolution $2^n \times 2^n$ consists of $2^n \times 2^n$ pixels each of which is assigned a value corresponding to its colour or grayness value.

- An Image is a **Function** $F : \Sigma^k \rightarrow \mathbb{R}$
  
  - **Gray Scale** $F(x) =$ grayness value
  
  - **Black-White** $F(x) \in \{0, 1\}$ (White, Black)

- The $2^n \times 2^n$ pixels can be considered to form a bound square on 2-D space with $x$ and $y$ as the orthogonal axes.

- The location of each of the pixel can be specified by a tuple, $(x, y)$, representing its $x$ and $y$ coordinate.

- The tuple $(x, y)$ is called the address of the pixel.

- The address tuple $(x, y)$ is such that $x \in [0, 2^n - 1]$ and $y \in [0, 2^n - 1]$ and hence we can specify the $x(y)$ coordinate as an $n$-bit binary number.

- The address of the pixel at $(x, y)$ is specified as a string $w \in \Sigma^n$ where $\Sigma = \{0, 1, 2, 3\}$.
Addressing of Pixels (Contd...)  

- If the $n$-bits representation of $x$ and $y$ coordinates are $x_{n-1}x_{n-2}\cdots x_1x_0$ and $y_{n-1}y_{n-2}\cdots y_1y_0$ respectively, then the address string $w = a_{n-1}a_{n-2}\cdots a_1a_0$ is such that $a_i \in \Sigma$ and $a_i = 2x_i + y_i, \forall i \in [0, 2^n - 1]$

![Diagram of addressable pixels]

Address of Black Square is 3022

Alternate Way of Getting Address

$$a_i = 2x_i + y_i$$

$$(18, 20)$$

$X = 10010$

$Y = 10100$

$A = 30120$
Black-White Images and Finite State Automata (FSA)

- The FSA representing the $2^n \times 2^n$ resolution black-white image is a non deterministic FSA, $A = (\Sigma, Q, \delta, I, F)$ where
  - $\Sigma$ is a finite set of alphabets, $\{0, 1, 2, 3\}$
  - $Q$ is a finite set of states
  - $\delta$ is a transition function as defined for a non-deterministic FSA.
  - $I \subseteq Q$ is a set of initial states. This can be equivalently represented as single initial state $q_0$ with $\epsilon$ transitions to all states in $I$
  - $F \subseteq Q$ is a set of final states
- The language recognized by $A$, $L(A) = \{w | w \in \Sigma^n, f(w) = 1\}$ i.e. the language recognized by the FSA consists of the addresses of the black pixels
Black-White Images and FSA (Contd ...)

\[ L = \{0,1,2,3\}^* \]

\[ L = \{0,3\}\{0,1,2,3\}^* \]

\[ L = \{0,1,2,3\}\{0,3\}\{0,1,2,3\}^* \]
Black-White Images and FSA (Contd ...)

- The same non-deterministic FSA of $m$ states can also be represented as follows
  - a row vector $I^A \in \{0, 1\}^{1 \times m}$ called the initial distribution
    
    \[ I_q^A = 1 \text{ if } q \text{ is an initial state, } 0 \text{ otherwise} \]
  - a column vector $F^A \in \{0, 1\}^{m \times 1}$ called the final distribution
    
    \[ F_q^A = 1 \text{ if } q \text{ is a final state, } 0 \text{ otherwise} \]
  - a matrix $W_a^A \in \{0, 1\}^{m \times m}$, $\forall a \in \Sigma$ called the transition matrix
    
    \[ W_{ap,q}^A = 1 \text{ if } q \in \delta(p, a), 0 \text{ otherwise} \]

- This FSA, $A$, defines the function $F : \Sigma^n \rightarrow \{0, 1\}$ by
  \[
  F(a_{n-1}a_{n-2} \cdots a_1a_0) = I^A \cdot W_{a_{n-1}}^A \cdot W_{a_{n-2}}^A \cdots W_{a_1}^A \cdot W_{a_0}^A \cdot F^A
  \]
  where the operation '.' indicates binary multiplication
Finite State Transducers (FSTs)

- An $m$ state finite state transducer FST from an alphabet $\Sigma_1$ to an alphabet $\Sigma_2$ is a Mealy Machine and is specified by
  - a row vector $I \in \{0, 1\}^{1 \times m}$ called the initial distribution,
  - a column vector $F \in \{0, 1\}^{m \times 1}$ called the final distribution and
  - binary matrices $W_{a,b} \in \{0, 1\}^{m \times m}$ $\forall a \in \Sigma_1$ and $b \in \Sigma_2$ called transition matrices

- In order to obtain a transformation of an image, we apply the corresponding FST to the FSA representing the image to get a new FSA. The application of an $n$ state FST to an $m$ state FSA $A = (I^A, F^A, W^A_\cdot, a \in \Sigma)$ produces an $n \times m$ state FSA $B = (I^B, F^B, W^B_\cdot, b \in \Sigma)$ as follows

$$I^B = I \otimes I^A$$

$$F^B = F \otimes F^A$$

$$W^B_{b} = \sum_{a \in \Sigma} W_{a,b} \otimes W^A_{a}, \forall b \in \Sigma$$

where the operation $\otimes$ is the ordinary tensor product of matrices.
FSTs (Contd ...)

- If $T$ and $Q$ are two matrices of size $s \times t$ and $p \times q$ then

$$
T \otimes Q = 
\begin{pmatrix}
T_{11}Q & \cdots & T_{1t}Q \\
\vdots & & \vdots \\
T_{s1}Q & \cdots & T_{st}Q
\end{pmatrix}
$$

- Example Finite State Transducer
Transformations on Black-White Images

- Translation through the $X$-Axis
  - by 1, 2, 4 pixel lengths

FST for translation by 1 unit

FST FOR TRANSLATION BY 2 UNITS

FST FOR TRANSLATION BY 4 UNITS
Transformations on Black-White Images

(Contd ...)

- by $\frac{1}{2}$ the original size ($2^{n-1}$ units) → change the most significant bit from 0 to 1 (1 to 0)

- by $\frac{1}{4}$ the original size ($2^{n-2}$ units) → replace the 2 most significant bits from 00 to 01, 01 to 10, 10 to 11 and 11 to 00

FST for translation by 1/2 square

FST for translation by 1/4 square
Transformations on Black-White Images

(Contd ...)

• Scaling

  – scale **up** by a factor of 2

  sub-square addressed as 03 in the original image becomes the bigger sub-square 0 in the scaled version. Similarly, the sub-squares 12, 21, 30 in the original image are scaled up to form the sub-squares 1, 2, 3 of the scaled image. Thus the FSA for the scaled up version can be obtained by introducing a new initial state which makes a 0(1, 2, 3) transition to the states reachable from the initial states of the original FSA by a path labeled 03(12, 21, 30). The formal construction is as follows.
Transformations on Black-White Images

(Contd ...)

Construction

Let $M = (\Sigma, Q, \delta, I, F)$ be the FSA representing a black and white image. The FSA, $M' = (\Sigma', Q', \delta', I', F')$ which represents the image scaled by a factor of 2 is constructed as follows.

1. $\Sigma' = \Sigma$.
2. $Q' = Q \cup \{q_0\}$, $I' = \{q_0\}$. $F' = F$
3. $\delta'(q_0, 0) = \delta(\delta(q_0, 0), 3)$
4. $\delta'(q_0, 1) = \delta(\delta(q_0, 1), 2)$
5. $\delta'(q_0, 2) = \delta(\delta(q_0, 2), 1)$
6. $\delta'(q_0, 3) = \delta(\delta(q_0, 3), 0)$
7. $\delta'(q, a) = \delta(q, a)$, $\forall a \in \Sigma$, $\forall q \in Q$
Transformations on Black-White Images

(Contd ...)

– scale down by a factor of 2

sub-square addressed as 0 in the original image becomes the smaller sub-square 03 in the scaled down version. Similarly the sub-squares 1, 2, 3 in the original image are scaled downed to form the sub-squares 12, 21, 30 in the scaled image. Thus the FSA for the scaled down version can be obtained by introducing new states. Any transition from an initial state labeled 0 is replaced by a path labeled 03. Similarly transitions from initial state labeled as 1, 2, 3 are replaced by paths labeled 12, 21, 30 by using the new states. The formal construction is as follows.
Transformations on Black-White Images
(Contd ...)

- Construction

Let $M = (\Sigma, Q, \delta, I, F)$ be the FSA representing a black-white image. The FSA, $M' = (\Sigma', Q', \delta', I', F')$ which represents the image scaled by a factor of 2 is constructed as follows

1. $\Sigma' = \Sigma$.
2. $Q' = Q \cup \{q'\} \cup \{q_0, q_1, q_2, q_3\}$, $I' = \{q'\}$, $F' = F$
3. $\delta'(q', 0) = \{q_0\}$, $\delta'(q_0, 3) = \{q|\exists i, i \in I, \delta(i, 0) = q\}$
4. $\delta'(q', 1) = \{q_1\}$, $\delta'(q_1, 2) = \{q|\exists i, i \in I, \delta(i, 1) = q\}$
5. $\delta'(q', 2) = \{q_2\}$, $\delta'(q_2, 1) = \{q|\exists i, i \in I, \delta(i, 2) = q\}$
6. $\delta'(q', 3) = \{q_3\}$, $\delta'(q_3, 0) = \{q|\exists i, i \in I, \delta(i, 3) = q\}$
7. $\delta'(q, a) = \delta(q, a)$, $\forall q \in Q$, $\forall a \in \Sigma$. 
Transformations on Black-White Images

(Contd ...)

SCALE BY FACTOR 2

SCALE BY FACTOR 1/2
Transformations on Black-White Images

(Contd ...)

• Rotation

– The rotation matrix for the clockwise rotation of angle $\theta$ about the coordinate axes is given by

\[
\begin{pmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{pmatrix}
\]

– Rotation through an angle of $45^\circ$ and scaled down by a factor of $\sqrt{2}$
Transformations on Black-White Images

(Contd ...)

Let $M = (\Sigma, Q, \delta, I, F')$ be the FSA representing the image. The operation of rotation involves the following 4 steps:

1. The FSA $M$ is scaled by a factor of $\frac{1}{2}$ to form the FSA $M'$

2. The transformation equations represented in terms of the old axes would be

$$x \leftarrow x - (y - 2^{n-1}) \quad y \leftarrow x + (y - 2^{n-1})$$

Thus we need to subtract $2^{n-1}$ from the $y$ coordinate. This can be achieved by changing the most significant bit of $y$ coordinate from 0 to 1 and 1 to 0. Hence we construct a FSA $M''$ from $M'$ as follows.

(a) $\Sigma'' = \Sigma', \quad Q'' = Q', \quad I'' = I', \quad F'' = F'$

(b) $\delta''(q', 0) = \delta'(q', 1)$

(c) $\delta''(q', 1) = \delta'(q', 0)$

(d) $\delta''(q', 2) = \delta'(q', 3)$
(e) \( \delta''(q', 3) = \delta'(q', 2) \)

(f) \( \delta''(q, a) = \delta'(q, a), \forall q \in Q', a \in \Sigma' \)

3. We apply the FST in the following figure to the FSA \( M'' \) to obtain the rotated FSA N. The FST in the following figure performs a transformation such that the x coordinate is replaced by x-y and y coordinate is replaced by x+y. The addition and subtraction can be done bit by bit using a four state FST where each state represents a carry over from previous calculation of 0 or 1 for each coordinate x and y. Since the \( i^{th} \) alphabet \( a_i = 2x_i + y_i \) we know both the x bit and the y bit at each stage. Hence the addition and subtraction can be done on them directly.

4. After shrinking the image, we have applied the rotation transformation. The rotation transformation has a scale factor of \( \sqrt{2} \) but the FST can only perform a one-one transformation. Hence there are pixels in the transformed image whose values have not been set. It is found that the transformation sets only those pixels whose addresses end in 0 or 3.
Transformations on Black-White Images

(Contd ...)

Hence in this step, we add transitions to the FSA N so that the pixels with address ending in 0 or 3 also, if they are black make the pixels whose addresses end in 1 or 2 also black. Hence

\[ \delta(q, 1) = \{ f | f \in F, f \in \delta(q, 0) \}, \forall q \in Q_N \]

\[ \delta(q, 2) = \{ f | f \in F, f \in \delta(q, 3) \}, \forall q \in Q_N \]

The FSA N constructed at the end of the fourth step represents the image rotated by an angle 45° and scaled by a factor of \( \frac{1}{\sqrt{2}} \).
Transformations on Black-White Images

(Contd ...)

FST for Rotation through $45^\circ$

$$
\begin{array}{ccc}
0 & 2 & 1 \\
1 & 1 & 1
\end{array}
$$

FST applied to Chess Board Fragment
Three Dimensional Objects and FSAs

- A solid object is considered to be a 3-dimensional array. Hence any point in the solid can be addressed as a 3-tuple \((x, y, z)\). For representing this solid object in the form of an FSA we simply have to extend the alphabet set of the FSA to \(\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}\). Any string \(w \in \Sigma^n\) gives the address of the point in the 3-dimensional space of size \(2^n \times 2^n \times 2^n\) enclosing the solid object. If the bit representation of the \(x\) coordinate is \(x_{n-1}x_{n-2} \cdots x_1x_0\), \(y\) coordinate is \(y_{n-1}y_{n-2} \cdots y_1y_0\), \(z\) coordinate is \(z_{n-1}z_{n-2} \cdots z_1z_0\), then the address of the square is the string \(w = a_{n-1}a_{n-2} \cdots a_1a_0\) such that

\[
a_i = 4x_i + 2y_i + z_i, \forall i \in [0, n - 1]\]
Three Dimensional Objects and FSAs

(Contd ...)

- Addressing scheme for three dimensional objects

\[ \Sigma = \{ 0, 1, 2, \ldots, 7 \} \]

\[ a_i = 4x_i + 2y_i + z_i \]

Example: Triangular Prism
Three Dimensional Objects and FSAs

(Contd ...)

- **Projection of 3D object on to \(XY, YZ, ZX\) planes**

  In order to visualize, the solid object has to be represented as a 2-dimensional image. A 2-dimensional image of a solid object can be obtained by projecting the solid onto a plane.

  Consider a black and white solid object. In order to form its projection on to the \(YZ\) plane, we have to suppress the \(X\) coordinate of the points. Thus if the FSA \(M = (\{0, 1, \cdots, 7\}, Q, \delta, I, F')\) represents the 3-D object then the FSA representing the image of the object projected onto the \(YZ\) plane is \(M' = (\{0, 1, 2, 3\}, Q, \delta', I, F')\) where

  \[
  \delta'(q, 0) = \delta(q, 0) \cup \delta(q, 4) \\
  \delta'(q, 1) = \delta(q, 1) \cup \delta(q, 5) \\
  \delta'(q, 2) = \delta(q, 2) \cup \delta(q, 6) \\
  \delta'(q, 3) = \delta(q, 3) \cup \delta(q, 7)
  \]
Three Dimensional Objects and FSAs

(Contd ...)

Similarly the projections can be formed on to the \(XY\) and \(ZX\) planes by suppressing the \(Z\) coordinate and the \(Y\) coordinate respectively.

The FSA representing the projection onto the \(XY\) plane is

\[
M'' = (\{0, 1, 2, 3\}, Q, \delta'', I, F)
\]

where

\[
\delta''(q, 0) = \delta(q, 0) \cup \delta(q, 1)
\]
\[
\delta''(q, 1) = \delta(q, 2) \cup \delta(q, 3)
\]
\[
\delta''(q, 2) = \delta(q, 4) \cup \delta(q, 5)
\]
\[
\delta''(q, 3) = \delta(q, 6) \cup \delta(q, 7)
\]

The FSA representing the projection onto the \(ZX\) plane is

\[
M''' = (\{0, 1, 2, 3\}, Q, \delta''', I, F)
\]

where

\[
\delta'''(q, 0) = \delta(q, 0) \cup \delta(q, 2)
\]
\[
\delta'''(q, 1) = \delta(q, 4) \cup \delta(q, 6)
\]
\[
\delta'''(q, 2) = \delta(q, 1) \cup \delta(q, 3)
\]
\[
\delta'''(q, 3) = \delta(q, 5) \cup \delta(q, 7)
\]
Three Dimensional Objects and FSAs

(Contd ...)

Projections of right angled prism

• Reconstruction of 3D objects from projections
  
  – Intersection of 3 orthogonal sweep patterns
    
    A solid object is often represented as 2-dimensional images on three mutually orthogonal planes (top-view, front-view, side-view). Hence it is useful to construct a solid object only from its projections. First we give an algorithm for constructing a 2-dimensional image from its projections on to X and Y axes. Then we extend this construction to obtain the 3-D object from its projections.
Three Dimensional Objects and FSAs

(Contd ...)

– Construction

Let \( M_1 = (\{0, 1\}, Q_1, \delta_1, I_1, F_1) \) be the FSA representing the \( X \)-axis projection and \( M_2 = (\{0, 1\}, Q_2, \delta_2, I_2, F_2) \) the \( Y \)-axis projection. Then the FSA \( N = (\{0, 1, 2, 3\}, Q, \delta, I, F) \) representing the 2-D image whose projections are \( M_1 \) and \( M_2 \) is given by

1. \( Q = Q_1 \times Q_2, I = I_1 \times I_2, F = F_1 \times F_2 \)

2. \( \delta([q_1, q_2], 0) = \delta_1(q_1, 0) \times \delta_2(q_2, 0) \)

3. \( \delta([q_1, q_2], 1) = \delta_1(q_1, 0) \times \delta_2(q_2, 1) \)

4. \( \delta([q_1, q_2], 2) = \delta_1(q_1, 1) \times \delta_2(q_2, 0) \)

5. \( \delta([q_1, q_2], 3) = \delta_1(q_1, 1) \times \delta_2(q_2, 1) \)

However, if one takes the projections of a 2-D image and reconstruct a 2-D image, the reconstructed image is not always same as the original image. In fact, the projections of any 2-D image on to the \( X \) or \( Y \) axis is a line segment. Hence the reconstructed image is always a rectangle. Thus only for rectangles, the reconstruction gives the original image.
Three Dimensional Objects and FSAs

(Contd ...)

The above restriction does not extend to 3-D objects. This is because when we consider the projections onto XY, YZ, ZX planes these projections are not independent of each other. When the 3D object is constructed, the 3 patterns (images) are swept along their normal axes (Z, X, Y respectively) to form 3 sweep objects and the intersection of these 3 objects represents the constructed solid. For example, if the projections on the XY, YZ, ZX planes are each circles of same radii. Then the 3 sweep objects are cylinders along Z, X, Y axes and the constructed solid would be the solid formed by the intersection of these 3 cylinders. Hence any object which can be obtained by sweeping patterns from 3 orthogonal planes can be constructed from its projections. The right angled prism is one such object.
Three Dimensional Objects and FSAs (Contd ...)

– Construction

If $M_{XY} = \{0, 1, 2, 3\}, Q_{XY}, \delta_{XY}, I_{XY}, F_{XY}$ represents the projection on the XY plane and $M_{YZ}$ and $M_{ZX}$ represent the projections on the YZ and ZX planes respectively, then the FSA $M = \{0, 1, \cdots, 7\}, Q, \delta, I, F$ representing the 3-dimensional object is as follows.

1. $Q = Q_{XY} \times Q_{YZ} \times Q_{ZX}$, $I = I_{XY} \times I_{YZ} \times I_{ZX}$, $F = F_{XY} \times F_{YZ} \times F_{ZX}$
2. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 0) = \delta_{XY}(q_{XY}, 0) \times \delta_{YZ}(q_{YZ}, 0) \times \delta_{ZX}(q_{ZX}, 0)$
3. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 1) = \delta_{XY}(q_{XY}, 0) \times \delta_{YZ}(q_{YZ}, 1) \times \delta_{ZX}(q_{ZX}, 2)$
4. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 2) = \delta_{XY}(q_{XY}, 1) \times \delta_{YZ}(q_{YZ}, 2) \times \delta_{ZX}(q_{ZX}, 0)$
5. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 3) = \delta_{XY}(q_{XY}, 1) \times \delta_{YZ}(q_{YZ}, 3) \times \delta_{ZX}(q_{ZX}, 2)$
6. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 4) = \delta_{XY}(q_{XY}, 2) \times \delta_{YZ}(q_{YZ}, 0) \times \delta_{ZX}(q_{ZX}, 1)$
7. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 5) = \delta_{XY}(q_{XY}, 2) \times \delta_{YZ}(q_{YZ}, 1) \times \delta_{ZX}(q_{ZX}, 3)$
8. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 6) = \delta_{XY}(q_{XY}, 3) \times \delta_{YZ}(q_{YZ}, 2) \times \delta_{ZX}(q_{ZX}, 1)$
9. $\delta([q_{XY}, q_{YZ}, q_{ZX}], 7) = \delta_{XY}(q_{XY}, 3) \times \delta_{YZ}(q_{YZ}, 3) \times \delta_{ZX}(q_{ZX}, 3)$
Three Dimensional Objects and FSAs

(Contd ...)

Right Angled Prism constructed from projections

The above figure gives the FSA constructed from the 3 projections of the right angled prism shown in the earlier figure. The constructed image here is equal to the original image. It should be noted that all solid objects cannot be constructed from their projections. The constructed object is the intersection of the three objects formed by sweeping the three projections along the perpendicular axes. Hence, only such objects can be reconstructed from their projections. Also, it should be noted that the projection of a hollow object will not indicate the hollowness of the object.
Weighted Finite Automata (WFA) and Gray-Scale Images

- A **weighted finite automaton** $M$ is specified by

  1. $Q$ a finite set of states.
  2. $\Sigma$ a finite set of alphabets.
  3. $W_\alpha : Q \times Q \to \mathbb{R}$ for all $\alpha \in \Sigma \cup \{\epsilon\}$, the weights of edges labeled $\alpha$.
  4. $I : Q \to (-\infty, \infty)$, the initial distribution.
  5. $F : Q \to (-\infty, \infty)$, the final distribution.

Here $W_\alpha$ is an $n \times n$ matrix where $n = |Q|$. $I$ is considered to be an $1 \times n$ row vector and $F$ is considered to be an $n \times 1$ column vector.
WFA and Gray-Scale Images (Contd ...)

- When representing the WFAs as figure, we follow a format similar to FSAs. Each state is represented by a node in a graph.

- The initial distribution and final distribution of each state is written as a tuple inside the state.

- A transition labeled $\alpha$ is drawn as a directed arc from state $p$ to $q$ if $W_\alpha(p, q) \neq 0$. The weight of the edge is written in brackets on the directed arc.

- We use the notation $I_q(F_q)$ to refer to the initial(final) distribution of state $q$. $W_\alpha(p, q)$ refers to the weight of the transition from $p$ to $q$. $W_\alpha(p)$ refers to the $p^{th}$ row vector of the weight matrix $W_\alpha$. It gives the weights of all the transitions from state $p$ labeled $\alpha$ in a vector form.

- Also $W_x$ refers to the product $W_{\alpha_1} \cdot W_{\alpha_2} \cdots W_{\alpha_k}$ where

$$x = \alpha_1 \alpha_2 \cdots \alpha_k$$
**WFA and Gray-Scale Images (Contd ...)**

- A WFA is said to be **deterministic** if its underlying FSA is deterministic.

- A WFA is said to be **ε-free** if the weight matrix $W_\varepsilon \equiv 0$ where 0 is the zero matrix of order $n \times n$.

- A WFA $M$ defines a function $F : \Sigma^* \rightarrow \mathbb{R}$, where for all $x \in \Sigma^*$ and $x = \alpha_1 \alpha_2 \cdots \alpha_k$,

$$F(x) = I \cdot W_{\alpha_1} \cdot W_{\alpha_2} \cdots W_{\alpha_k} \cdot F$$

where the operation $\cdot$ is matrix multiplication.

- A **path** $P$ of length $k$ is defined as a tuple $(q_0 q_1 \cdots q_k, \alpha_1 \alpha_2 \cdots \alpha_k)$ where $q_i \in Q, 0 \leq i \leq k$ and $\alpha_i \in \Sigma, 1 \leq i \leq k$ such that $\alpha_i$ denotes the label of the edge traversed while moving from $q_{i-1}$ to $q_i$.

- The **weight** of a path $P$ is defined as

$$W(P) = I_{q_0} \cdot W_{\alpha_1}(q_0, q_1) \cdot W_{\alpha_2}(q_1, q_2) \cdots W_{\alpha_k}(q_{k-1}, q_k) \cdot F_{q_k}$$
WFA and Gray-Scale Images (Contd ...)

• The function $F : \Sigma^* \rightarrow \mathbb{R}$ represented by a WFA $M$ can be equivalently defined as follows

$$F(x) = \sum_{P \text{ is a path of } M \text{ labeled } x} W(P), x \in \Sigma^*.$$  

• A function $F : \Sigma^* \rightarrow \mathbb{R}$ is said to be average preserving if

$$F(w) = \frac{1}{m} \sum_{\alpha \in \Sigma} F(w\alpha)$$  

for all $w \in \Sigma^*$ where $m = |\Sigma|$.

• A WFA $M$ is said to be average preserving if the function that it represents is average preserving.

• WFA $M$ is average preserving if and only if

$$\sum_{\alpha \in \Sigma} W_{\alpha} \cdot F = mF,$$

where $m = |\Sigma|$.  

WFA and Gray-Scale Images (Contd ...)

- A WFA is said to be **i-normal** if the initial distribution of every state is 0 or 1 i.e. $I_{q_i} = 0$ or $I_{q_i} = 1$ for all $q_i \in Q$

- A WFA is said to be **f-normal** if the final distribution of every state is 0 or 1 i.e. $F_{q_i} = 0$ or $F_{q_i} = 1$ for all $q_i \in Q$

- A WFA is said to be **I-normal** if there is only one state with non-zero initial distribution

- A WFA is said to be **F-normal** if there is only one state with non-zero final distribution
WFA and Gray-Scale Images (Contd ...)

- Representation of Gray-Scale Images
  - A gray-scale digital image of finite resolution consists of $2^m \times 2^m$ pixels, where each pixel takes a real grayness value (in reality the value ranges as 0, 1, ::, 256)
  - By a multi-resolution image, we mean a collection of compatible $2^m \times 2^m$ resolution images for $n \geq 0$
  - We can define our finite resolution image as a function $F_I : \Sigma^k \rightarrow \Re$, where $F_I(x)$ gives the value of the pixel at address $x$.
  - A multi-resolution image is a function $F_I : \Sigma^* \rightarrow \Re$. It is shown that for compatibility, the function $F_I$ should be average preserving i.e.
    $$F_I(x) = \frac{1}{4}[F_I(x0) + F_I(x1) + F_I(x2) + F_I(x3)]$$
  - A WFA $M$ is said to represent a multi-resolution image if the function $F_M$ represented by $M$ same as the function $F_I$ of the image
WFA and Gray-Scale Images (Contd ...)

Example: WFA computing linear grayness function

**Example 1** Consider the 2 state WFA shown in figure. The $I = (1, 0)$ and $F = (\frac{1}{2}, 1)$ and the weight matrices are $W_0 = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix}$,

$W_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 0 & 1 \end{pmatrix}$ $W_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 0 & 1 \end{pmatrix}$ $W_3 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}$.

Then we can calculate the values of pixels as follows. $F(03) =$ sum of weights all paths labeled $03$.

$$F(03) = 1 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot 1 = \frac{1}{8} + \frac{1}{4} = \frac{5}{8}$$

similarly for $f(123)$ we have $f(123) = \frac{1}{16} + \frac{1}{8} + \frac{1}{8} + \frac{1}{4} = \frac{9}{16}$. The images obtained by this WFA are shown for resolutions $2 \times 2$, $4 \times 4$ and $128 \times 128$ in the above figure.
Inference and De-Inference Algorithms

- De-Inferencing
  - Given a WFA $M$, $(I, F, W_0, W_1, W_2, W_3)$ we need to construct a finite resolution approximation of the multi-resolution image represented by $M$
  - Let the image to be constructed be $I$ of resolution $2^k \times 2^k$.
  - Then for all $x \in \Sigma^k$, we have to compute $F(x) = I \cdot W_x \cdot F$
  - The algorithm is as follows. The algorithm computes $\phi_p(x)$ or $p \in Q$ for all $x \in \Sigma^i, 0 \leq i \leq k$. Here $\phi_p$ is the image of state $p$
Inference and De-Inference Algorithms

(Contd ...)

Algorithm 1 De_Infer_WFA

Input : WFA $M = (I, F, W_0, W_1, W_2, W_3)$.
Output : $f(x)$, for all $x \in \Sigma^k$.

begin

Step 1 : Set $\phi_p(\epsilon) \leftarrow F_p$ for all $p \in Q$

Step 2 : For $i = 1, 2, \cdots, k$, do the following

begin

Step 3 : For all $p \in Q$, $x \in \Sigma^{i-1}$ and $\alpha \in \Sigma$ compute

$$\phi_p(\alpha x) \leftarrow \sum_{q \in Q} W_a(p, q) \cdot \phi_q(x)$$

end for

Step 4 : For each $x \in \Sigma^k$, compute

$$f(x) = \sum_{q \in Q} I_q \cdot \phi_q(x).$$

Step 5 : Stop.

end
Inference and De-Inference Algorithms

(Contd ...)

- The time complexity of the above algorithm is $O(n^2 4^k)$, where $n$ is the number of states in the WFA and $4^k = 2^k \cdot 2^k$ is the number of pixels in the image.

- We know that $f(x)$ can be computed either by summing the weights of all the paths labeled $x$ or by computing $I \cdot W_x \cdot F$.

- Finding all paths labeled of length $k$ takes $k \cdot (4k)^n$ time. Since $n \gg k$ we prefer the matrix multiplication over this.

- **Inferencing**

  - Let $\mathcal{I}$ be the digital gray-scale image of finite resolution $2^k \times 2^k$. The inference algorithm which is an iterative algorithm obtains the WFA $M$ representing the image $\mathcal{I}$.

  - In the algorithm:

    * $N$ is the index of the last state created

    * $i$ is the index of the first unprocessed state

    * $\phi_p$ is the image represented by state $p$
Inference and De-Inference Algorithms

(Contd ...)

* $F_x$ represents the sub-image at the sub-square labeled $x$, while $f_{avg}(x)$ represents the average pixel value of the sub-image at the sub-square labeled $x$

* $\gamma: Q \leftarrow \Sigma^*$ is a mapping of states to sub-squares

Algorithm 2 Infer-WFA

Input : Image $\mathcal{I}$ of size $2^k \times 2^k$.
Output : WFA $M$ representing image $\mathcal{I}$.

begin
Step 1 : Set $N \leftarrow 0$, $i \leftarrow 0$, $F_{q_0} \leftarrow f_{avg}(\epsilon)$, $\gamma(q_0) \leftarrow \epsilon$.
Step 2 : Process $q_i$, i.e., for $x = \gamma(q_i)$ and each $\alpha \in \{0,1,2,3\}$ do

    begin
    Step 3 : If there are $c_0, c_1, \ldots, c_N$ such that $f_{wa} = c_0 \phi_0 + c_1 \phi_1 + \cdots + c_N \phi_N$ where $\phi_j = f_{\gamma(q_j)}$ for $0 \leq j \leq N$ then set $W_\alpha(q_i, q_j) \leftarrow c_j$ for $0 \leq j \leq N$.
    Step 4 : else set $\gamma(q_{N+1}) \leftarrow x \alpha$, $F_{q_{N+1}} \leftarrow f_{avg}(x \alpha)$, $W_\alpha(q_i, q_{N+1}) \leftarrow 1$ and $N \leftarrow N + 1$.
    end
    Step 5 : Set $i \leftarrow i + 1$ and goto Step 2.
    Step 6 : Set $I_{q_0} \leftarrow 1$ and $I_{q_j} \leftarrow 0$ for all $1 \leq j \leq N$.
end
Inference and De-Inference Algorithms

(Contd ...)

Example 2 Consider the linearly sloping ap-function $f$ introduced in example 1. Let us apply the inference algorithm to find a minimal ap-WFA generating $f$.

First, the state $q_0$ is assigned to the square $\epsilon$ and we define $F_{q_0} = \frac{1}{2}$. Consider then the four sub-squares 0, 1, 2, 3. The image in the first sub-square 0 can be expressed as $\frac{1}{2} \cdot f$(it is obtained from the original image by decreasing the gray-scale by $\frac{1}{2}$) so that we define $W_0(q_0, q_0) = 0.5$.

The image in sub-square 1 cannot be expressed as a linear combination of $f$ so that we have to use a second state $q_1$. Define $W_1(q_0, q_1) = 1$ and $F_{q_1} = \frac{1}{2}$.(the average grayness of sub-square 1 is $\frac{1}{2}$). Let $f_1$ denote the image in sub-square 1 of $f$.

The image in sub-square 2 is the same as in sub-square 1, so that $W_2(q_0, q_1) = 1$. In the quadrant 3 we have image which can be expressed as $2 \cdot f_1 - \frac{1}{2} \cdot f$. We define $W_3(q_0, q_0) = -\frac{1}{2}$ and $W_3(q_0, q_1) = 2$. The out going transitions from state $q_0$ are now ready.

Consider then the images in the squares 10, 11, 12, and 13. They can be expressed as $f_1 - \frac{1}{4} \cdot f$, $\frac{3}{2} \cdot f_1 - \frac{1}{2} \cdot f$, $\frac{3}{2} \cdot f_1 - \frac{1}{2} \cdot f$, and $2 \cdot f_1 - \frac{3}{4} \cdot f$. This gives us the ap-WFA shown in figure 1. The initial distribution is $(1, 0)$.
It is been shown that, in Algorithm 2, each state is independent of the other state, i.e., \( \phi_i \), where \( 1 \leq i \leq n \) cannot be expressed as a linear combination of other states.

\[
c_1\phi_1 + c_2\phi_2 + \cdots + c_n\phi_n = 0
\]

implies that \( c_i = 0 \) for all \( 1 \leq i \leq n \). Hence the WFA obtained by Algorithm 2 is a minimum state WFA.

Now consider Step 3 of Algorithm 2. This step asks for finding \( c_0, c_1, \ldots, c_n \) such that \( f_\alpha = c_0\phi_0 + c_1\phi_1 + \cdots + c_n\phi_n \), i.e., to express the sub-image of the sub-square \( x\alpha \) as a linear combinations of the images represented by the states so far created. Let the size of the sub image be \( 2^k \times 2^k \). Then the above equation can be restated as follows

\[
I_{x\alpha,2^k\times2^k} = c_0 \cdot I_{0,2^k\times2^k} + c_1 \cdot I_{1,2^k\times2^k} + \cdots + c_n \cdot I_{n,2^k\times2^k}
\]

where \( I_{q_i,2^k\times2^k} \) is the \( 2^k \times 2^k \) image represented by the state \( q_i \) and \( I_{x\alpha,2^k\times2^k} \) is the \( 2^k \times 2^k \) sub-image at the sub-square addressed by \( x\alpha \). The equations can be rewritten as follows.

\[
\begin{align*}
c_0I_0(1,1) + c_1I_1(1,1) + \cdots + c_nI_n(1,1) &= I_{x\alpha}(1,1) \\
c_0I_0(1,2) + c_1I_1(1,2) + \cdots + c_nI_n(1,2) &= I_{x\alpha}(1,2) \\
\vdots & \vdots \\
c_0I_0(1,2^k) + c_1I_1(1,2^k) + \cdots + c_nI_n(1,2^k) &= I_{x\alpha}(1,2^k) \\
\vdots & \vdots \\
c_0I_0(2^k,1) + c_1I_1(2^k,1) + \cdots + c_nI_n(2^k,1) &= I_{x\alpha}(2^k,1) \\
c_0I_0(2^k,2) + c_1I_1(2^k,2) + \cdots + c_nI_n(2^k,2) &= I_{x\alpha}(2^k,2) \\
\vdots & \vdots \\
c_0I_0(2^k,2^k) + c_1I_1(2^k,2^k) + \cdots + c_nI_n(2^k,2^k) &= I_{x\alpha}(2^k,2^k)
\end{align*}
\]

This is nothing but a set of linear equations. It can be rewritten in matrix form as \( A \cdot C = B \), where \( A \) is a \( 4^k \times (n + 1) \) matrix where the \( i^{th} \) column represents the \( 2^k \times 2^k = 4^k \) pixels of the image \( I_i \) represented by state \( q_i \). \( C \) is an \((n + 1) \times 1\) column vector of the coefficients. \( B \) is a \( 4^k \times 1 \) vector containing the pixels of the image \( I_{x\alpha} \). Thus the Step 3 reduces to solving a set of linear equations.
One well known method of attacking this problem is using the **Gaussian Elimination** technique. But for using this technique, the rank of the matrix $A$ should be $\min(4^k, n)$. But in the general case, $\text{rank}(A)$ is found to be $\leq \min(4^k, n)$. Hence this method cannot be used in our case.

Another standard method for solving a set of linear equations is **Singular Value Decomposition**. This method not only gives the solution if it exists, but also in the case of non-existing solution gives us the least mean square approximate solution. The computed coefficients are such that $\|B - AC\|$ is minimum, where $\|M\| = \sqrt{M^T M}$.

**Approximate Representation**

In image processing applications, it is not always required that the images be exactly represented. We can see that by introducing an error parameter in the above algorithm, the number of states required for representation can be reduced. While solving the linear equations in Step 3 of Algorithm 2, we get a solution with least mean square error. We can accept the solution if this error is less than a positive quantity $\delta > 0$. This way we can represent the image approximately with a smaller automaton.

**Compression**

We also see that if the resultant WFA can be stored with lesser file space than the image, then we can get a compressed representation of the image. Also any WFA can be made $f$-normal. Further we can make our WFA $I$-normal since we need not bother about representing images of size 0. Further it was observed that in most cases, the weight matrices obtained are sparse. Hence, we need to store only the weights on the edges in a file.
Observations from Implementation

We have applied the inference algorithm for four images of size $256 \times 256$. The figure shows the compression obtained for the images in cases where the error factor was equal to 0%, 5%, 10% and 15%. The reconstructed images are also shown in figure. It can be observed that while error up to 10% does not disturb the figure much, using error of 15% distorts the figure pretty badly. Also it was observed that the number of states obtained depends on the regularity of the image itself.

We believe that the image can be further compressed by a smarter way of storing the WFA in a file. Currently for each edge, 4 bytes are needed to store the weight (type float). On an average, an $n$ state WFA has $4 \frac{n^2}{2} = 2n^2$ edges. The number of bytes used to store an $n$ state WFA is $4(2n^2) = 8n^2$. In order to obtain compression of say 50% for an image of size $2^k \times 2^k$.

\[
8n^2 \leq \frac{4^k}{2} \\
\Rightarrow n^2 \leq \frac{4^k}{16} = 4^{k-2} \\
\Rightarrow n \leq 2^{k-2}
\]

For a $256 \times 256$ image, $n$ should be less than 64 in order to obtain any good compression.
Inference and De-Inference Algorithms

(Contd ...)

• Implementation

  – De-Inference

    * \( \forall x \in \Sigma^k \), calculate \( f(x) = IW_{o_1} \cdots W_{o_k} F \).

  – Inference

    * Singular Value Decomposition

  – Compression (↑)

    * Number of states (↓) in WFA
    * Number of edges (↓) in WFA

    * Error (↑) for approximate representation.

    * Regularity (↑) of image
Inference and De-Inference Algorithms

(Contd ...)

Observations
Incremental Inference of WFAs

- Image $\mathcal{I}_2$ very close to Image $\mathcal{I}_1$
- Example successive frames of a movie
- WFA of $\mathcal{I}_1$ already known
- Find and Infer only **dissimilar** sub-squares
- Cut-Paste the inferred WFA on WFA of $\mathcal{I}_1$
Operation Cut-Paste

- Find all paths labeled address of sub-square
- Remove the incoming edges by duplication
- Cut last-link and append target WFA
Cooperating Distributed Weighted Finite Automata (CDWFA)

- A Cooperating Distributed Weighted Finite Automata with $n$-components, $n$-WFA is a 5-tuple $\Gamma = (Q, \Sigma, W_\alpha, I, F)$ where,
  
  1. $Q$ is an $n$-tuple $(Q_1, Q_2, \ldots, Q_n)$ where each $Q_i$ is the set of states corresponding to the $i^{th}$ component
  
  2. $\Sigma$ is the finite set of alphabet
  
  3. $W_\alpha$ is an $n$-tuple $(W_\alpha^1, W_\alpha^2, \ldots, W_\alpha^n)$ of weight matrices (weights of edges labeled $\alpha$ for each $\alpha \in \Sigma \cup \{\epsilon\}$ where each

$$W_\alpha^i : Q_\text{union} \times Q_\text{union} \rightarrow \mathbb{R}, \ 1 \leq i \leq n$$

  4. $I : Q_\text{union} \rightarrow (-\infty, \infty)$ is the initial distribution
  
  5. $F : Q_\text{union} \rightarrow (-\infty, \infty)$ is the final distribution

where $Q_\text{union} = \bigcup_i Q_i$

- Each of the component WFA of the $n$-WFA is of the form $M_i = (Q_i, \Sigma, W_\alpha^i)$, $1 \leq i \leq n$. Note that here $Q_i$'s need not be disjoint

- Each of $W_\alpha^i$ is an $m \times m$ matrix where $m = |Q_\text{union}|$ and these matrices are sparse matrices
CDWFA (Contd ...)

- $I$ is considered to be an $1 \times m$ row vector
- $F$ is considered to be an $m \times 1$ column vector
- A $n$-WFA is said to be **deterministic** if each of its component WFA is deterministic
- A $n$-WFA $\Gamma$ defines a function $f : \Sigma^* \rightarrow \Re$, where for all $x \in \Sigma^*$ and $x = \alpha_1\alpha_2\ldots\alpha_k$,

$$f(x) = \sum(I \cdot W^{i_1} \cdot W^{i_2} \cdot \ldots \cdot W^{i_k} \cdot F)$$

- The summation is over all possible paths for $x$ in the various components, the operation `$\cdot$' is the matrix multiplication and $1 \leq i_1, i_2, \ldots, i_k \leq n$ refers to the different possible components in which the $n$-WFA can be in, while reading the input string $x$.
- A **path $P$** of length $k$ of an $n$-WFA $\Gamma$ is a tuple $(q_0q_1 \ldots q_k, \alpha_1\alpha_2\ldots\alpha_k)$ where $q_i \in Q_{\text{union}}, 0 \leq i \leq k$ and $\alpha_i \in \Sigma, 1 \leq i \leq k$ such that $\alpha_i$ denotes the label of the edge traversed while moving from the state $q_{i-1}$ to the state $q_i$. 
CDWFA (Contd ...)

- The **weight** of a path $P$ of an $n$-WFA $\Gamma$ is defined as

$$W(P) = I_{q_0} \cdot W^i_{\alpha_1}(q_0, q_1) \cdot W^i_{\alpha_2}(q_1, q_2) \cdot \ldots \cdot W^i_{\alpha_k}(q_{k-1}, q_k) \cdot F_{q_k}$$

where $1 \leq i_1, i_2, \ldots, i_k \leq n$.

The function $f : \Sigma^* \rightarrow \mathbb{R}$ represented by the $n$-WFA $\Gamma$ can be equivalently defined as follows

$$f(x) = \sum_{P \text{ is a path of } \Gamma \text{ labeled } x} W(P), \quad x \in \Sigma^*.$$

- A function $f : \Sigma^* \rightarrow \mathbb{R}$ is said to be **average preserving** if

$$f(w) = \frac{1}{m} \sum_{\alpha \in \Sigma} f(w\alpha)$$

for all $w \in \Sigma^*$ where $m = |\Sigma|$.

- An $n$-WFA $\Gamma$ is said to be **average preserving** if the function that it represents is average preserving.

- An $n$-WFA, $\Gamma$, is said to be **i-normal** if the initial distribution of every state is 0 or 1 i.e. $I_{q_i} = 0$ or $I_{q_i} = 1$ for all $q_i \in Q_{\text{union}}$.

- An $n$-WFA, $\Gamma$, is said to be **f-normal** if the final distribution of every state is 0 or 1 i.e. $F_{q_i} = 0$ or $F_{q_i} = 1$ for all $q_i \in Q_{\text{union}}$. 

Kamala Krithivasan  
Dept. of C.S.E.  
IITM
CDWFA (Contd ...)

- An $n$- WFA, $\Gamma$, is said to be **I-normal** if there is only one state with non-zero initial distribution

- An $n$- WFA, $\Gamma$, is said to be **F-normal** if there is only one state with non-zero final distribution

- **Modes of Acceptance**

  - **$t$-mode acceptance**: The automaton which has a state with the non-zero initial distribution begins the processing of the input string. Suppose that the system starts from the component $i$. In the component $i$ the system follows its transition function given by it’s weight matrix $W^i_\alpha$ as any “stand alone” WFA. The control is transfered from the component $i$ to component $j$ only if the system arrives at a state $q \not\in Q_i$ and $q \in Q_j$. The selection of $j$ is nondeterministic if $q$ belongs to more than one $Q_j$. This process is repeated and we accept the string if the system after reading the entire string reaches any one of the states which has a non-zero final state distribution. It does not matter in which component the system is in
CDWFA (Contd ...)

- The instantaneous description of the \( n \)-WFA (ID) in the \( t \)-mode is given by a 3-tuple \( (q, w, i) \) where \( q \in Q_{\text{union}}, w \in \Sigma^*, 1 \leq i \leq n \)

- In this ID of the \( n \)-WFA, \( q \) denotes the current state of the whole system, \( w \) the portion of the input string yet to be read and \( i \) the index of the component in which the system is currently in

- The transition between the ID’s is defined as follows

  1. \( (q, aw, i) \vdash (q', w, i) \) iff \( W_a^i(q, q') \neq 0 \) where \( q \in Q_i, q' \in Q_{\text{union}}, a \in \Sigma \cup \{\epsilon\}, w \in \Sigma^*, 1 \leq i \leq n \)

  2. \( (q, w, i) \vdash (q, w, j) \) iff \( q \in Q_j - Q_i \)

Let \( \vdash^* \) be the reflexive and transitive closure of \( \vdash \)

- The language accepted by the \( n \)-WFA \( \Gamma = (Q, \Sigma, W_\alpha, I, F') \)
  working in \( t \)-mode is defined as follows,

\[
L_t(\Gamma) = \left\{ w \in \Sigma^* \mid \begin{array}{c}
(q_0, w, i) \vdash^* (q_f, \epsilon, j) \text{ for some } q_f \text{ with non-zero final distribution}, 1 \leq j, i \leq n \text{ and } q_0 \in Q_i \\
also f_\Gamma(w) = \text{ weight of the string } w \text{ and } f_\Gamma(w) > 0
\end{array} \right\}
\]
CDWFA (Contd …)

- ***-mode acceptance**: The automaton which has a state with the non-zero initial distribution begins the processing of the input string. Suppose the system starts the processing from the component \(i\). Unlike the termination mode (\(t\)-mode), here there is no restriction. The automaton can transfer the control to any other component at any time if possible, i.e., if there is some \(j\) such that \(q \in Q_j\) then the system can transfer the control to the component \(j\). The selection is done non-deterministically if there is more than one \(j\). The instantaneous description and the language accepted by the system in *-mode can be defined analogously. The language accepted in *-mode is denoted as \(L_\ast(\Gamma)\)
CDWFA (Contd ...)

- $k$-mode (≤ $k$-mode, ≥ $k$-mode) acceptance: The component which has a state with the non-zero initial distribution begins the processing of the input string. Suppose the system starts the processing from the component $i$. The system transfers the control to the other component $j$ only after the completion of exactly $k(k'(k' \leq k), k'(k' \geq k))$ number of steps in the component $i$, i.e., if there is a state $q \in Q_j$ then the transition from component $i$ to the component $j$ takes place only if the system has already completed $k(k'(k' \leq k), k'(k' \geq k))$ steps in component $i$. If there is more than one choice for $j$ the selection is done non-deterministically.

- All the 5 modes of acceptance for CDWFA are equivalent
Representation of Gray-Scale Images using

\(n\)-WFA

\[
\begin{array}{c}
\begin{array}{c}
0,1,2,3(1/2) \\
1,2(1/4) \\
3(1/2)
\end{array}
\end{array}
\]

2-WFA computing linear grayness function

- Consider the 2-WFA shown in the figure. The \(I = (1, 0)\) and \(F = (\frac{1}{2}, 1)\) and the weight matrices corresponding to the 2 components are as follows:

\[
\begin{align*}
W_0^1 &= \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix}, \\
W_1^1 &= \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ 0 & 0 \end{pmatrix}, \\
W_2^1 &= \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ 0 & 0 \end{pmatrix}, \\
W_3^1 &= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \end{pmatrix}, \\
W_0^2 &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \\
W_1^2 &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \\
W_2^2 &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}
\end{align*}
\]

and \(W_3^2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\). In the figure the dotted lines correspond to the change in the control from one component to another. Then we can calculate the values of pixels as follows. \(f(13) = \text{sum of weights of all paths labeled 13.}\)

\[
f(13) = 1 \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} + 1 \times \frac{1}{2} \times \frac{1}{2} \times 1 + 1 \times \frac{1}{2} \times 1 \times 1 = \frac{1}{8} + \frac{1}{4} + \frac{1}{4} = \frac{5}{8}
\]

similarly for \(f(123)\) we have \(f(123) = \frac{1}{16} + \frac{1}{8} + \frac{1}{4} = \frac{9}{16}\). The images obtained by this 2-WFA are shown for resolutions 2 \(\times\) 2, 4 \(\times\) 4 and 128 \(\times\) 128 in the above figure.

- Though the number of matrices in the \(n\)-WFA are more than the usual WFA the advantage of using the \(n\)-WFAs is that most of the matrices are sparse matrices and thus the matrix computations are much faster than in the usual WFA case.
Inference and De-Inference Algorithms for $n$-WFA

- Inferencing

Let $I$ be a digital gray-scale(colour) multiresolution image given by the average preserving function $f : \Sigma^* \rightarrow \mathbb{R}$. We construct an average preserving $(2^m \times 2^m)$-WFA $M$ such that $f_M = f$. During the construction:

- The states created in various components are represented as a tuple $[q_i, j]$ where $[q_i, j]$ denotes the $i^{th}$ state in the $j^{th}$ component $1 \leq j \leq 2^m \times 2^m$;
- $[q_0, 0]$ is the initial state corresponding to the whole image which acts as a distribution centre i.e. it distributes the whole image into various sub-images for the components to process;
- $N$ is the index of the last state created in the corresponding component $j$, $1 \leq j \leq 2^m \times 2^m$;
- $i$ is the index of the first unprocessed state in corresponding component $j$, $1 \leq j \leq 2^m \times 2^m$;
- $\gamma : Q_{\text{union}} \rightarrow \Sigma^*$ is a mapping of the states to subsquares, where $Q_{\text{union}}$ denotes the union of all the states in the $2^m \times 2^m$ components and the initial state $[q_0, 0]$;
- $\phi_{[q_p, j]}$ is the image represented by the state $[q_p, j]$ and
- $f_w$ represents the subimage at the subsquare labeled $w$
Inference and De-Inference Algorithms for $n$-WFA (Contd ...)

**Input**: Image $\mathcal{I}$ given by an average preserving function, $f : \Sigma^* \rightarrow \mathbb{R}$ and $2^m \times 2^m$-the number of components of the $(2^m \times 2^m)$-WFA to be constructed

**Output**: $(2^m \times 2^m)$-WFA $M$ representing the image $\mathcal{I}$

**Begin**

1. Set $N \leftarrow 0, i \leftarrow 0$, for each of the components $j$, $1 \leq j \leq 2^m \times 2^m$,
   \[ F([q_0,0]) = f(\varepsilon) \text{ and } \gamma([q_0,0]) \leftarrow \varepsilon \]

2. Create new states $[q_0,j]$ for $1 \leq j \leq 2^m \times 2^m$ such that $F([q_0,j]) = f(\omega)$ and $\gamma([q_0,j]) = \omega$, $\omega \in \Sigma^m$ $1 \leq j \leq 2^m \times 2^m$ where $j = (\omega)_4 + 1$ i.e. $j = \text{value of } \omega \text{ in base } 4 + 1$. The components are numbered from 1 to $2^m \times 2^m$ and the subimages are numbered from 0 to $(2^m \times 2^m) - 1$. So if the picture is divided into $2^m \times 2^m$ sub-images, then the sub-image with address $\omega \in \{0,1,2,3\}^*$ will be given by the component $j = (\omega)_4 + 1$.

3. For each component $j, 1 \leq j \leq 2^m \times 2^m$ the following steps are carried out parallelly
   (a) Process $[q_i,j]$, i.e. for $x = \gamma([q_i,j])$ and each $\alpha \in \{0,1,2,3\}$ do
      \begin{enumerate}
      \item If there are $c_0, c_1, \ldots, c_N$ such that
        \[ f_{\alpha l} = c_0 \phi_0 + c_1 \phi_1 + \cdots + c_N \phi_N, \text{ where } \phi_l = f_{\gamma([q_i,j])} \text{ for } l, 1 \leq l \leq N \]
        then set
        \[ W_i^j([q_i,j],[q_i,j]) \leftarrow q_1, \text{ for } 0 \leq l \leq N \]
      \item else set
        \[ \gamma([q_{i+1},j]) \leftarrow x\alpha, F_{[q_{i+1},j]} \leftarrow f(x) \]
        \[ W_{i+1}^j([q_i,j],[q_{i+1},j]) \leftarrow 1 \text{ and } N \leftarrow N + 1 \]
      \end{enumerate}
   (b) Set $i \leftarrow i + 1$, if $i \leq N$ then go to Step (a)

4. Set $I([q_0,0]) = 1, I([q_i,j]) = 0$ for $i = 1, \ldots, N, 1 \leq j \leq 2^m \times 2^m$, where $I$ is the initial distribution of $M$

**End**

Kamala Krithivasan
Dept. of C.S.E., IITM
De-Inferencing

- Given a \((2^m \times 2^m)\)-WFA \(M(I, F, W_0^j, W_1^j, W_2^j, W_3^j), 1 \leq j \leq 2^m \times 2^m\) and we want to construct a finite resolution approximation of the multiresolution image represented by \(M\)
- Let the image to be constructed be \(I\) of resolution \(2^k \times 2^k\)
- Then for all \(x \in \Sigma^k\), we have to compute \(f(x) = \sum I \cdot W_x \cdot F\)
- The de-inference algorithm computes \(\phi_{[q_p,j]}(x)\) for all \(p, j\) and for all \(x \in \Sigma^i, 0 \leq i \leq k\). Here \(\phi_{[q_p,j]}\) is the image of state \([q_p, j]\)

\[\text{Input : WFA } M = (I, F, W_0^j, W_1^j, W_2^j, W_3^j), 1 \leq j \leq 2^m \times 2^m.\]

\[\text{Output : } f(x), \text{ for all } x \in \Sigma^k.\]

\[\text{begin}\]

For all \(j, 1 \leq j \leq 2^m \times 2^m\) paralee do

1. Set \(\phi_{[q_p,j]}(\epsilon) \leftarrow F_{[q_p,j]}\) for all \(p, j\)

2. For \(i = 1, 2, \cdots, k - m\), do the following

   \[\text{begin}\]

3. For all \(p, j, x \in \Sigma^i\) and \(\alpha \in \Sigma\) compute

   \[\phi_{[q_p,j]}(\alpha x) \leftarrow \sum_{[q_r,j], \text{r varies}} W_\alpha([q_p,j], [q_r,j]) \cdot \phi_{[q_p,j]}(x),\]

   \[\text{end for}\]

4. For each \(x \in \Sigma^{k-m}\), compute

   \[f(x) = \sum_{[q_r,j], \text{r varies}} I_{[q_r,j]} \cdot \phi_{[q_r,j]}(x).\]

5. Stop

end

Kamala Krithivasan \hspace{1cm} Dept. of C.S.E. \hspace{1cm} IITM
Inference and De-Inference Algorithms for $n$-WFA (Contd ...)

- **Compression**
  - If the resultant $(2^m \times 2^m)$-WFA can be stored with lesser file space than the image, then we get a compressed representation of the image.
  - We have observed that in most cases, the weight matrices obtained for the various components are sparse. Hence, we need to store only the non-zero weights in the file.
  - The size of the given image (in pixels form) and the size of the inferred $n$-WFA are used as a measure for calculating the compression.
  - Suppose the given image is of size $x$ bytes ($2^k \times 2^k$ resolution) and the size of the inferred $n$-WFA is say $y$ bytes. Then the percentage of compression is calculated as follows,

\[
Compression = \frac{(x - y) \times 100}{x}
\]

- We have applied the inference algorithm for various gray-scale and colour images of sizes $64 \times 64, 128 \times 128, 256 \times 256$ and $512 \times 512$. The original images and the reconstructed images with various percent of errors are shown. Here we take $m$ to be 2 and hence the number of components are $2^2 \times 2^2 = 16$.

- In the case of colour images the RGB values of the pixels are stored in the $n$-WFA and while de-inferencing the RGB values corresponding to the pixels are extracted from the $n$-WFA to get the colour image.
Figure 2: Original and Reconstructed Image with 5% error

Figure 3: Original and Reconstructed Images with 5, 10 and 15% errors

Figure 4: Original and Reconstructed gray-scale and colour Images with 10% errors
Inference and De-Inference Algorithms for

\( n\)-WFA (Contd ...)

Compression Statistics

<table>
<thead>
<tr>
<th>Image</th>
<th>Compression rate</th>
<th>Number of states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( n)-WFA</td>
<td>WFA</td>
</tr>
<tr>
<td>Original image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reconstructed image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with 5% error</td>
<td>16.14%</td>
<td>22.7%</td>
</tr>
</tbody>
</table>

Table 1: The compression statistics for Figure 1

<table>
<thead>
<tr>
<th>Image</th>
<th>Compression rate</th>
<th>Number of states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( n)-WFA</td>
<td>WFA</td>
</tr>
<tr>
<td>Original image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reconstructed image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with 5% error</td>
<td>26.12%</td>
<td>25.86%</td>
</tr>
<tr>
<td>Reconstructed image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with 10% error</td>
<td>47.92%</td>
<td>47.19%</td>
</tr>
<tr>
<td>Reconstructed image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with 15% error</td>
<td>65.33%</td>
<td>62.83%</td>
</tr>
</tbody>
</table>

Table 2: The compression statistics for Figure 2
Inference and De-Inference Algorithms for 

$n$-WFA (Contd ...)

<table>
<thead>
<tr>
<th>Image</th>
<th>Compression rate</th>
<th>Number of states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n$-WFA WFA</td>
<td>$n$-WFA WFA</td>
</tr>
<tr>
<td>Original gray-scale image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reconstructed gray-scale image with 10% error</td>
<td>55.14% 59.86%</td>
<td>367 95</td>
</tr>
<tr>
<td>Original colour image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reconstructed colour image</td>
<td>54.80%</td>
<td>1106 -</td>
</tr>
</tbody>
</table>

Table 3: The compression statistics for Figure 3

<table>
<thead>
<tr>
<th>Image</th>
<th>Run-time (in Seconds) of our de-inferencing algorithm</th>
<th>Run-time (in Seconds) of the existing de-inferencing algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconstructed image of baby with 5% error</td>
<td>6</td>
<td>129</td>
</tr>
<tr>
<td>Reconstructed image of baby with 10% error</td>
<td>4</td>
<td>98</td>
</tr>
<tr>
<td>Reconstructed image of baby with 15% error</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>Reconstructed gray-scale image of Taj Mahal with 10% error</td>
<td>70</td>
<td>1394</td>
</tr>
</tbody>
</table>

Table 4: The comparison in the run-time of our de-inference algorithm with the classical one
Transformations on Digital Images

- Almost every transformation of an image involves moving (scaling) pixels or changing grayness (colour) values between the pixels. These image transformations can be specified by Weighted Finite Transducers.

- An $n$-state Weighted Finite Transducer (WFT) $M$ from an alphabet $\Sigma = \{0, 1, 2, 3\}$ into the alphabet $\Sigma$ is specified by

  1. weight matrices $W_{a,b} \in \mathbb{R}^{n \times n}$ for all $a \in \Sigma \cup \{\epsilon\}$ and $b \in \Sigma \cup \{\epsilon\}$,

  2. a row vector $I \in \mathbb{R}^{1 \times n}$, called the initial distribution, and

  3. a column vector, $F \in \mathbb{R}^{n \times 1}$, called the final distribution.

- The WFT $M$ is called $\epsilon$-free if the weight matrices $W_{\epsilon,\epsilon}$, $W_{a,\epsilon}$, and $W_{\epsilon,b}$ are zero matrices for all $a \in \Sigma$ and $b \in \Sigma$. 
Transformations on Digital Images (Contd ..)

- The WFT M defines a function $f_M : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}$, called weighted relation between $\Sigma^*$ and $\Sigma^*$, by

$$f_M(u, v) = I \cdot W_{u,v} \cdot F,$$

for all $u \in \Sigma^*, v \in \Sigma^*$

where

$$W_{u,v} = \sum_{a_1 \cdots a_k, b_1 \cdots b_k} W_{a_1, b_1} \cdot W_{a_2, b_2} \cdots \cdot W_{a_k, b_k},$$

if the sum converges. (If the sum does not converge, $f_M(u, v)$ remains undefined). The sum is taken over all decompositions of $u$ and $v$ into symbols $a_i \in \Sigma \cup \{\epsilon\}$ and $b_i \in \Sigma \cup \{\epsilon\}$, respectively

- In the special case of $\epsilon$-free transducers,

$$f_M(a_1a_2 \ldots a_k, b_1b_2 \ldots b_k) = I \cdot W_{a_1, b_1} \cdot W_{a_2, b_2} \cdots \cdot W_{a_k, b_k} \cdot F,$$

for $a_1a_2 \ldots a_k \in \Sigma^k, b_1b_2 \ldots b_k \in \Sigma^k$, and $f_M(u, v) = 0$, if $|u| \neq |v|$
Transformations on Digital Images (Contd ..)

- Recall that an \( n \)-WFA defines a multiresolution function
  \[
  f : \Sigma^* \longrightarrow \mathbb{R}
  \]
  where for all \( x \in \Sigma^* \) and \( x = \alpha_1 \alpha_2 \ldots \alpha_k \),
  \[
  f(x) = \sum (I \cdot W_{\alpha_1}^{i_1} \cdot W_{\alpha_2}^{i_2} \ldots \cdot W_{\alpha_k}^{i_k} \cdot F)
  \]
  where the summation is over all possible paths for \( x \) in the various components, the operation \( ' \cdot ' \) is the matrix multiplication and \( 1 \leq i_1, i_2, \ldots, i_k \leq n \) refers to the different possible components in which the \( n \)-WFA can be in, while reading the input string \( x \).

- Let \( \rho : \Sigma^* \times \Sigma^* \longrightarrow \mathbb{R} \) be a weighted relation and \( f : \Sigma^* \longrightarrow \mathbb{R} \) a multiresolution function represented by an \( n \)-WFA. The application of \( \rho \) to \( f \) is the multiresolution function
  \[
  g = \rho(f) : \Sigma^* \longrightarrow \mathbb{R}
  \]
  over \( \Sigma \) defined by
  \[
  g(v) = \sum_{u \in \Sigma^*} f(u)\rho(u, v), \text{ for all } v \in \Sigma^*, \quad (1)
  \]
  provided the sum converges. The application \( M(f) \) of WFT \( M \) to \( f \) is defined as the application of the weighted relation \( f_M \) to \( f \), i.e.
  \[
  M(f) = f_M(f)
  \]
Transformations on Digital Images (Contd ..)

• Equation (1) defines an application of a WFT to an image in the pixel form. When the image is available in the $n$-WFA-compressed form we can apply a WFT directly to it and compute the regenerated image again from the transformed $n$-WFA

• The application of an $\epsilon$-free $n$-state WFT $M$ to an $m$-state $k$-WFA $\Gamma$ over the alphabet $\Sigma$ specified by initial distribution $I^\Gamma$, final distribution $F^\Gamma$ and weight matrices $W_\alpha^\Gamma, \alpha \in \Sigma$, is the $mn$-state $k$-WFA $\Gamma' = M(\Gamma)$ over the alphabet $\Sigma$ with initial distribution $I^{\Gamma'} = I \otimes I^\Gamma$, final distribution $F^{\Gamma'} = F \otimes F^\Gamma$ and weight matrices

$$W_b^{\Gamma'} = \sum_{a \in \Sigma} W_{a,b} \otimes W_a^\Gamma,$$

for all $b \in \Sigma$. 
Transformations on Digital Images (Contd ..)

- Here, $\otimes$ denotes the ordinary tensor product of matrices (called also Kronecker product or direct product), defined as follows: Let $T$ and $Q$ be matrices of sizes $s \times t$ and $p \times q$, respectively. Then their tensor product is the matrix

$$T \otimes Q = \begin{pmatrix} T_{11}Q & \cdots & T_{1t}Q \\ \vdots & & \vdots \\ T_{s1}Q & \cdots & T_{st}Q \end{pmatrix}$$

of size $sp \times tq$

- clearly $f_{\Gamma'} = M(f_{\Gamma})$, i.e. the multiresolution function defined by $\Gamma'$ is the same as the application of the WFT $M$ to the multiresolution function computed by the WFA $\Gamma$

- We note that every WFT $M$ is a linear operator $\mathbb{R}^{\Sigma^*} \rightarrow \mathbb{R}^{\Sigma^*}$. In other words,

$$M(r_1f_1 + r_2f_2) = r_1M(f_1) + r_2M(f_2),$$

for all $r_1, r_2 \in \mathbb{R}$ and $f_1, f_2 : \Sigma^* \rightarrow \mathbb{R}$. More generally, any weighted relation acts as a linear operator
Transformations on Digital Images (Contd ..)

- Scaling

  - Consider a gray-scale image of resolution $2^n \times 2^n$. We consider the scaling with respect to the center of the image. So the coordinate axes are shifted to the center of the image. Thus the new coordinate axes are $x' = x - 2^{n-1}$ and $y' = y - 2^{n-1}$
  
  - The operation of scaling by a factor $k$ is defined as follows,

    $$x' \leftarrow \frac{x'}{k} \quad y' \leftarrow \frac{y'}{k}$$

  - The operation of scaling up by a factor of 4 and scaling down by a factor of 1/4 is illustrated in Figure 5

![Figure 5: Scaling of a gray-scale image]
Transformations on Digital Images (Contd ..)

– It is can be seen that the subsquare addressed as 033 in the original gray-scale image becomes the bigger subsquare 0 in the scaled version of the gray-scale image.

– Similarly, the subsquares 122,211 and 300 in the original image are scaled up to form the subsquares 1,2 and 3 in the scaled version of the gray-scale image.

– The $n$-WFA for the scaled version of the gray-scale image can be obtained from the $n$-WFA representing the original gray-scale image by introducing a new initial state $q'_0$ which on reading a 0 (1,2,3) makes a transition to the states reachable from the initial states of the original $n$-WFA by a path labeled 033 (122,211,300).

– The formal construction of the $n$-WFA representing the scaled version of the gray-scale image is as follows.
Transformations on Digital Images (Contd ..)

– Construction

Let $\Gamma = (Q, \Sigma, W_\alpha, I, F)$ be an $I$-normal $n$-WFA representing the original gray-scale image with $q_0 \in Q_i$ for some $i, 1 \leq i \leq n$ with non-zero initial distribution.

The $I$-normal $n$-WFA, $\Gamma' = (Q', \Sigma', W'_\alpha, I', F')$, representing the scaled version of the original gray-scale image by a factor of 4, is constructed as follows,

1. $Q' = (Q'_1, Q'_2, \ldots, Q'_n)$ where $Q'_i = Q_i \cup \{q'_0\}$ and $Q'_j = Q_j$
   for all $j, 1 \leq j \leq n, j \neq i$

2. $\Sigma' = \Sigma$

3. $W'_\alpha = (W'^1_\alpha, W'^2_\alpha, \ldots, W'^n_\alpha)$ where

   (a) $W'^{ij}_\alpha(q'_0, r) = \sum (W'^{ij}_\alpha(q_0, p) \cdot W'^{jk}_\alpha(p, q) \cdot W'^{kj}_\alpha(q, r))$
   for all $p, q, r \in Q_{\text{union}}(= \cup_i Q_i)$ and for all $j, k 1 \leq j, k \leq n$

   (b) $W'^{ij}_\alpha(p, q) = W'^{ij}_\alpha(p, q)$ for all $p, q \in Q_{\text{union}}$

   (c) $W'^{ij}_\alpha(q'_0, r) = \sum (W'^{ij}_\alpha(q_0, p) \cdot W'^{jk}_\alpha(p, q) \cdot W'^{kj}_\alpha(q, r))$
   for all $p, q, r \in Q_{\text{union}}(= \cup_i Q_i)$ and for all $j, k 1 \leq j, k \leq n$

   (d) $W'^{ij}_1(p, q) = W'^{ij}_1(p, q)$ for all $p, q \in Q_{\text{union}}$

   (e) $W'^{ij}_2(q'_0, r) = \sum (W'^{ij}_2(q_0, p) \cdot W'^{jk}_1(p, q) \cdot W'^{kj}_1(q, r))$
   for all $p, q, r \in Q_{\text{union}}(= \cup_i Q_i)$ and for all $j, k 1 \leq j, k \leq n$

   (f) $W'^{ij}_2(p, q) = W'^{ij}_2(p, q)$ for all $p, q \in Q_{\text{union}}$

   (g) $W'^{ij}_3(q'_0, r) = \sum (W'^{ij}_3(q_0, p) \cdot W'^{jk}_0(p, q) \cdot W'^{kj}_0(q, r))$
   for all $p, q, r \in Q_{\text{union}}(= \cup_i Q_i)$ and for all $j, k 1 \leq j, k \leq n$

   (h) $W'^{ij}_3(p, q) = W'^{ij}_3(p, q)$ for all $p, q \in Q_{\text{union}}$

4. $I'$, the initial distribution is such that $I'(q'_0) = 1$ and $I'(q) = 0$
   for all $q \in Q_{\text{union}}(= \cup_i Q_i)$

5. $F'$, the final distribution is same as that of the original $n$-WFA
   i.e., $F'(q'_0) = F(q_0)$ and $F'(q) = F(q)$ for all $q \in Q_{\text{union}}$.

– Figure 6(B) illustrates the above construction
Transformations on Digital Images (Contd ..)

(A) 2-WFA representing the original grayscale image

(B) 2-WFA representing the grayscale image scaled by a factor of 4

(C) 2-WFA representing the grayscale image scaled by a factor 1/4

Figure 6: 2-WFA for scaling the given gray-scale image
Transformations on Digital Images (Contd ..)

- The operation of scaling of a gray-scale image by a factor of \( \frac{1}{4} \) is also illustrated in Figure 6(C)

- When the subsquare is scaled by a factor of \( \frac{1}{4} \) the subsquare addressed as 0 in the original image becomes the smaller subsquare 033 in the scaled version

- Similarly the subsquares addressed as 1, 2, 3 in the original image are scaled down to form the subsquares 122, 211, 300 respectively in the scaled image

- The \( n \)-WFA for the scaled down version of the gray-scale image can be obtained by introducing new states

- Any transition from an initial state labeled 0 is replaced by a path labeled 033

- Similarly transitions from initial state labeled as 1, 2, 3 are replaced by paths labeled 122, 211, 300 by using the new states

- The formal construction is as follows
Transformations on Digital Images (Contd ..)

– Construction

Let $\Gamma = (Q, \Sigma, W, I, F)$ be an $n$-WFA representing the original gray-scale image with $q_0 \in Q_i$ for some $i, 1 \leq i \leq n$ with non-zero initial distribution. The $I$ normal $n$-WFA, $\Gamma' = (Q', \Sigma', W', I', F')$, representing the scaled down version of the gray-scale image by a factor of $\frac{1}{4}$ is constructed as follows,

1. $Q' = (Q'_1, Q'_2, \ldots, Q'_n)$ where $Q'_i = Q_i \cup \{q'_0, q'_1, q'_2, q'_3\} \cup \{q'_{03}, q'_{12}, q'_{21}, q'_{30}\}$ and $Q'_j = Q_j$ for all $j, 1 \leq j \leq n, j \neq i$

2. $\Sigma' = \Sigma$

3. $W'_a = (W'^1_a, W'^2_a, \ldots, W'^n_a)$ where

(a) $W'^0_i(q'_0, q'_0) = W'^i_i(q'_0, q'_1) = W'^ii_i(q'_0, q'_2) = W'^iii_i(q'_0, q'_3) = 1$,

(b) $W'^i_0(q'_0, q'_{03}) = W'^i_1(q'_1, q'_{12}) = W'^i_2(q'_2, q'_{21}) = W'^i_3(q'_3, q'_{30}) = 1$,

(c) $W'^i_3(q'_{03}, q) = W'^i_3(q_0, q), W'^i_2(q'_{12}, q) = W'^i_2(q_0, q)$,

(d) $W'^i_1(q'_{21}, q) = W'^i_1(q_0, q), W'^i_0(q'_{30}, q) = W'^i_0(q_0, q)$, for all $q \in Q_{union}$.

(e) $W'^i_0(q'_{12}, q) = W'^i_1(q'_{12}, q) = W'^i_2(q'_{12}, q) = 0$ for all $q \in Q_{union}$.

(f) $W'^i_0(q'_{21}, q) = W'^i_1(q'_{21}, q) = W'^i_2(q'_{21}, q) = 0$ for all $q \in Q_{union}$.

(g) $W'^i_0(q'_{30}, q) = W'^i_1(q'_{30}, q) = W'^i_2(q'_{30}, q) = 0$ for all $q \in Q_{union}$.

(h) $W'^i_0(q, p) = W'^i_0(q, p), W'^i_1(q, p) = W'^i_1(q, p)$,

and for all $i, 1 \leq i \leq n$. 

Kamala Krithivasan
Dept. of C.S.E.
IITM
Transformations on Digital Images (Contd ..)

4. \( I'(q') = I(q_0) \) and \( I'(q) = 0 \)
   for all other \( q \in Q/Q_i = \cup_i Q_i \)

5. \( F'(q') = F'(q_0) = F'(q_1) = F'(q_2) = F'(q_3) = F'(q_0) = F'(q_12)F'(q_2)F'(q_30) = 0 \) and \( F'(q) = F(q) \) for all \( q \in Q/Q_i \).

- The colour images in RGB format are stored in 24 bits per pixel with one byte each for the value of the three primary colours, namely, red, green and blue.

- In representing the colour image using the \( n \)-WFA we use three functions one each corresponding to the three primary colours.

- So the transformation of scaling for colour images is same as that mentioned for the gray-scale images except that the \( n \)-WFA corresponding to the colour image, defines three functions one each for the three primary colours.

- We illustrate the scaling up of the colour image by a factor of 4 and the scaling of the colour image by a factor of \( \frac{1}{4} \) using an example in Figure 7.
Transformations on Digital Images (Contd ..)

Original Image

Scale factor of 4

Scale factor of 1/4

Figure 7: Scaling of the colour image
Transformations on Digital Images (Contd ..)

- Translation
  - Let $\Gamma$ be an $n$-WFA representing a gray-scale image of resolution $2^n \times 2^n$. Suppose that we want to translate the image from left to right, along the $x$-axis by one pixel, then the image wraps around on translation i.e. the pixels in the $(2^n - 1)^{th}$ column are moved to the $0^{th}$ column
  - This is equivalent to adding 1 to the $x$ coordinate of the pixels. For example if the $n$-bit $x$ coordinate of the pixel is $w01^r$, then the $x$ coordinate of this pixel after the translation would be $w10^r$, $0 \leq r \leq n - 1$
  - The $y$ coordinate of the pixel remains unchanged
  - The weighted finite transducer for this translation is given in Figure 8
  - The $n$-WFA, $\Gamma'$ representing the translated image can be obtained from the $n$-WFA, $\Gamma$ by applying the WFT to $\Gamma$
Transformations on Digital Images (Contd ..)

![Diagram of transformations](image-url)

Figure 8: Translation by 1 unit

- To translate the gray-scale image by 2 units, we have to add 2 to the \( x \) coordinate of each pixel. This is the same as adding 1 to the binary representation of \( x \) coordinate after ignoring the 0th bit.

- In a similar manner the translation by 4 units is achieved by adding 1 to the \( x \) coordinate of each pixel after ignoring both the 0th and 1st bits.

- The WFTs for these operations are given in Figure 9.
Transformations on Digital Images (Contd ..)

Figure 9: Translation by 2,4 units

- if we want to translate the gray-scale image by half the original size i.e. by $2^{n-1}$ units, then we have to change the most significant bit from 0 to 1 or from 1 to 0

- Similarly if we have to translate the image by one fourth of its original size i.e. by $2^{n-2}$ units, then we have to replace the two most significant bits of the $x$ coordinate from 00 to 01, 01 to 10, 10 to 11, 11 to 00

- The Figure 10 gives the weighted finite transducers for these operations
Transformations on Digital Images (Contd ..)

Figure 10: Translation by 1/2 and 1/4 of square

- In order to perform a translation on the colour image we have to apply the corresponding WFT on all the three functions represented by the $n$-WFA

- We also illustrate the operation of translation on colour images with the following example in Figure 11
Transformations on Digital Images (Contd ..)

Figure 11: Translation of colour image by 1/2 and 1/4 of square

• Rotation

  – we show how rotation of gray-scale images through an angle of $\theta$ can be performed using the weighted finite transducers

  – We consider the coordinate axes to be shifted to the center of the gray-scale image so that the rotation would be in the anti-clockwise direction about the center of the image
Transformations on Digital Images (Contd ..)

- The rotation matrix for the clockwise rotation of angle $\theta$ about the coordinate axes is given by
  \[
  \begin{pmatrix}
  \cos\theta & -\sin\theta \\
  \sin\theta & \cos\theta
  \end{pmatrix}
  \]
- we discuss the case of rotation through an angle of $45^\circ$
- Let $x, y$ be the address of the pixel in the original coordinate system of the gray-scale image $x'$ and $y'$ represent the address of the pixel in the new coordinate system
- Then the rotation by an angle of $45^\circ$ can be specified as follows
  \[
  x'' \leftarrow \frac{x' - y'}{\sqrt{2}} \quad y'' \leftarrow \frac{x' + y'}{\sqrt{2}}
  \]
- The division by $\sqrt{2}$ is not easily achievable by the FST. So we scale the rotated image by a factor of $\frac{1}{\sqrt{2}}$ so as to give the following transformation
  \[
  x'' \leftarrow \frac{x' - y'}{2} \quad y'' \leftarrow \frac{x' + y'}{2}
  \]
- This not only rotates the given gray-scale image by an angle of $45^\circ$ but also scales the image by a factor of $\frac{1}{\sqrt{2}}$
Transformations on Digital Images (Contd ..)

Let $\Gamma = (Q, \Sigma, W_\alpha, I, F)$ be the $n$-WFA representing the original gray-scale image. The operation of rotation of the gray-scale image by $45^\circ$ and scaling by a factor of $\frac{1}{\sqrt{2}}$ is given by the following steps

1. The $n$-WFA $\Gamma$ is first scaled by a factor of $\frac{1}{2}$ to get the $n$-WFA $\Gamma' = (Q', \Sigma, W'_\alpha, I', F')$.

2. The transformation equations given in terms of the old axes would be

$$x_1 \leftarrow x - (y - 2^{n-1}) \quad y_1 \leftarrow x + (y - 2^{n-1})$$

where $(x_1, y_1)$ is obtained from $(x, y)$ after performing the following operations: scaling the given image, shifting the axes to the centre of the image, rotating the image by an angle of $45^\circ$, and then shifting back to the old axes. So we first need to subtract $2^{n-1}$ from the $y$ coordinate. This can be easily achieved by changing the most significant bit of the $y$ coordinate from 0 to 1 and 1 to 0. Hence we construct the $n$-WFA $\Gamma'' = (Q'', \Sigma, W''_\alpha, I'', F'')$ from $\Gamma'$ as follows.
Figure 12: WFT for rotation by 45°

(a) $Q'' = Q', I'' = I', F'' = F'$

(b) $W''_0(q', p) = W'_1(q', p)$

(c) $W''_1(q', p) = W'_0(q', p)$

(d) $W''_2(q', p) = W'_3(q', p)$

(e) $W''_3(q', p) = W'_2(q', p)$

(f) $W''_\alpha(q, p) = W'_\alpha(q, p) \forall q, p \in Q', \alpha \in \Sigma$
We apply the WFT in Figure 12 to the $n$-WFA $\Gamma''$ to obtain the rotated $n$-WFA $\Gamma'''$. The WFT in the Figure 12 transforms the $x$ coordinate to $x - y$ and the $y$ coordinate to $x + y$. The addition and subtraction can be done bit by bit using a four state WFT wherein each state represents a carry over from the previous calculation of 0 or 1 for each coordinate $x$ and $y$. Since the $i^{th}$ alphabet $a_i = 2x_i + y_i$, we know both the $x$ and $y$ bit at each stage. Hence the addition and subtraction can be done on them directly.

3. After scaling the image by a factor of $\frac{1}{\sqrt{2}}$ we have applied the rotation transformation. The rotation transformation has a scale factor of $\frac{1}{\sqrt{2}}$ but the WFT can only perform a one-one transformation. Hence there are pixels in the transformed image whose values have not been set. It is found that the transformation sets only pixels whose addresses end in 0 or 3. Hence in this final step we add transitions to the $n$-WFA $\Gamma'''$ so that the pixels with address ending with 0 or 3 will set their value to pixels whose addresses end with 1 or 2.
Hence

\[ W_1'''(q, P) = W_0'''(q, p), \text{ for all } p \text{ with non-zero final distribution} \]

and for all \( q \in Q''' \)

\[ W_2'''(q, P) = W_3'''(q, p), \text{ for all } p \text{ with non-zero final distribution} \]

and for all \( q \in Q''' \)

The \( n \)-WFA \( \Gamma''' \) constructed at the end of the fourth step represents the image rotated by an angle 45° and scaled by a factor of \( \frac{1}{\sqrt{2}} \)

- The rotation of colour images can be done in a similar manner as that of the gray-scale images. We illustrate the rotation of colour images by an angle of 45° in Figure 13.

![Figure 13: Rotation of colour image by an angle of 45°](image-url)
Transformations on Three Dimensional Objects

- Addressing scheme for 3D objects

  - A solid object is considered to be a three dimensional array.
    Hence any point in the solid can be addressed as a 3-tuple $(x, y, z)$

  - In order to represent the three dimensional solid object in the form of a FSA we have to extend the alphabet set $\Sigma$, of the FSA to $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$

  - Now any string $w \in \Sigma^n$ gives the address of a point in the three dimensional space of size $2^n \times 2^n \times 2^n$ enclosing the solid object

  - If the bit representation of the $x$ coordinate is $x_{n-1}x_{n-2}\ldots x_1x_0$, $y$ coordinate is $y_{n-1}y_{n-2}\ldots y_1y_0$, $z$ coordinate is $z_{n-1}z_{n-2}\ldots z_1z_0$, then the address of the point is the string

    $w = a_{n-1}a_{n-2}\ldots a_1a_0$ such that $a_i = 4x_i + 2y_i + z_i$, $\forall i \in [0, n - 1]$

Kamala Krithivasan
Dept. of C.S.E.
IITM
Transformations on Three Dimensional Objects (Contd ...)

– The Figure 14 shows how the 8 sub-cubes a of cube are addressed. The Figure also gives the FSA which generates the right angled prism

– Whenever we say size of the object we refer to the three dimensional space enclosed by the three dimensional object

![Diagram of three dimensional object and FSA](image)

Figure 14: Addressing scheme and an example automaton
Transformations on Three Dimensional Objects (Contd ...)

- Scaling of three dimensional objects
  - For a three dimensional object of resolution $2^n \times 2^n \times 2^n$, we consider the scaling with respect to the center of the object i.e. we shift the coordinate axes to the center of the object
  - The new coordinate axes are defined by $x' = x - 2^{n-1}$, $y' = y - 2^{n-1}$, $z' = z - 2^{n-1}$
  - The operation of scaling the object by a factor is defined as follows,
    $$x' \leftarrow \frac{x'}{k}, \quad y' \leftarrow \frac{y'}{k}, \quad z' \leftarrow \frac{z'}{k}$$
  - The operation of scaling a three dimensional object by a factor of 2 is illustrated in Figure 15
Transformations on Three Dimensional

Objects (Contd ...)

![Scale factor of 2]

![Scale factor of 1/2]

Figure 15: Illustration of 3D scaling

- It can be seen that the subsquares addressed as 03, 47 in the original three dimensional object become the bigger subsquares 0 and 4 in the scaled up version of factor 2

- Similarly, the subsquares 12, 56, 21, 65, 30, 74 in the original three dimensional object are scaled up to form the subsquares 1, 5, 2, 6, 3, 7
Transformations on Three Dimensional Objects (Contd ...)

– Thus the FSA for the scaled up version of the object can be obtained by introducing a new initial state \( q'_0 \) which makes a \( 0(1,2,3,4,5,6,7) \) transition to the states reachable from the initial states of the original FSA by a path labeled \( 03(12,21,30,47,56,65,74) \)

– The formal construction of the required FSA representing the scaled up object with a scale factor of 2 is as follows

– **Construction**

Let \( M = (\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}, Q, \delta, I = \{q_0\}, F) \) be an FSA representing the three dimensional object. The FSA, \( M' = (\Sigma', Q', \delta', I', F') \) which represents the scaled up three dimensional object by a factor of 2 is constructed as follows,

1. \( \Sigma' = \Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\} \)
2. \( Q' = Q \cup \{q'_0\}, I' = \{q'_0\}, F' = F \)
3. \( \delta'(q'_0, 0) = \delta(\delta(q_0, 0), 3) \)
4. \( \delta'(q'_0, 1) = \delta(\delta(q_0, 1), 2) \)
5. \( \delta'(q'_0, 2) = \delta(\delta(q_0, 2), 1) \)
6. \( \delta'(q'_0, 3) = \delta(\delta(q_0, 3), 0) \)
7. \( \delta'(q'_0, 4) = \delta(\delta(q_0, 4), 7) \)
8. \( \delta'(q'_0, 5) = \delta(\delta(q_0, 5), 6) \)
9. \( \delta'(q'_0, 6) = \delta(\delta(q_0, 6), 5) \)
10. \( \delta'(q'_0, 7) = \delta(\delta(q_0, 7), 4) \)
Transformations on Three Dimensional Objects (Contd ...)

– The Figure 16(B) illustrates this construction

(A) FSA representing the prism

(B) FSA representing the prism scaled by a factor of 2

(C) FSA representing the prism scaled by a factor of 1/2

Figure 16: FSA representing the scaled versions of the prism
Transformations on Three Dimensional Objects (Contd ...)

- The operation of scaling down a three dimensional object by a factor of $\frac{1}{2}$ is also illustrated in Figure 16(C).

- When the three dimensional object is scaled down a factor of $\frac{1}{2}$, the subsquare addressed as 0 in the three dimensional object becomes the subsquare 03 in the scaled down version of the three dimensional object.

- Similarly the subsquares addressed as 1,2,3,4,5,6,7 become the subsquares 12,21,30,47,56,65,74 in the scaled version of the three dimensional object.

- The FSA for the scaled down version of the three dimensional object can be obtained from the original FSA by introducing new states.

- Any transition from an initial state labeled 0 in the original FSA is replaced by a path labeled 03.
Transformations on Three Dimensional Objects (Contd ...)

- Similarly transitions from initial state labeled as 1,2,3,4,5,6,7 are replaced by paths labeled 12,21,30,47,56,65,74 using the new states

- The formal construction of the required FSA representing the scaled down version of the three dimensional object by a factor of $\frac{1}{2}$ is given below

- **Construction**

  Let $M = (\Sigma, Q, \delta, I, F)$ be the FSA representing the three dimensional object. The FSA, $M' = (\Sigma', Q', \delta', I', F')$ representing the scaled down version of the three dimensional object by a factor of $\frac{1}{2}$ is constructed as follows.

  1. $\Sigma' = \Sigma$
  2. $Q' = Q \cup \{q'_1\}$
  3. $\delta'(q'_1, 0) = \{q'_0\}, \delta'(q'_1, 3) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 0) = q\}$
  4. $\delta'(q'_1, 1) = \{q'_1\}, \delta'(q'_1, 2) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 1) = q\}$
  5. $\delta'(q'_1, 2) = \{q'_2\}, \delta'(q'_2, 1) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 2) = q\}$
  6. $\delta'(q'_1, 3) = \{q'_3\}, \delta'(q'_3, 0) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 3) = q\}$
  7. $\delta'(q'_1, 4) = \{q'_4\}, \delta'(q'_4, 7) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 4) = q\}$
  8. $\delta'(q'_1, 5) = \{q'_5\}, \delta'(q'_5, 6) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 5) = q\}$
  9. $\delta'(q'_1, 6) = \{q'_6\}, \delta'(q'_6, 5) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 6) = q\}$
  10. $\delta'(q'_1, 7) = \{q'_7\}, \delta'(q'_7, 4) = \{q \mid \exists q_i, q_i \in I, \delta(q_i, 7) = q\}$
  11. $\delta'(q, a) = \delta(q, a), \forall q \in Q, \forall a \in \Sigma$
Transformations on Three Dimensional Objects (Contd ...)

- Translation of three dimensional objects
  
  - Let the finite state automata, $M$, represent a three dimensional object enclosing a three dimensional space of size $2^n \times 2^n \times 2^n$. Consider the translation along the $y$-axis from left to right by one pixel
  
  - We assume that the object is wrapped around on translation i.e. the pixels in the $(2^n - 1)$ column are moved to the $0^{th}$ column
  
  - This translation is equivalent to adding 1 to the $y$ coordinate of each pixel. If the $n$-bit $y$ coordinate of pixel is $w01^r$ then the address coordinate of this pixel after translation would be $w10^r$, $0 \leq r \leq n - 1$
  
  - There is no change in the value of the $x$ and $z$ coordinates
  
  - The finite state transducer for this translation is found in Figure 17. The FSA $M'$ representing the translated image can be obtained from $M$ by applying this FST to $M$
In the FST, the transformation in a bit of \( y \) coordinate from 0 to 1 (1 to 0) would be represented as 0/2, 1/3, 4/6 and 5/7 (2/0, 3/1, 6/4 and 7/5)

to translate the image by 2 units, we have to add 2 to the \( y \) coordinate of each pixel. This is same as adding 1 to the binary representation of the \( y \) coordinate after ignoring the \( 0^{th} \) bit

Similarly a translation by 4 units can be achieved by adding 1 to the \( y \) coordinate after ignoring both the \( 0^{th} \) and \( 1^{st} \) bits

The FST for these operations are shown in Figure 17

Figure 17: FST’s for translation of the three dimensional object
Transformations on Three Dimensional Objects (Contd ...)

- To translate the object by half the size of the object i.e. by $2^{n-1}$ units, we have to simply change the most significant bit from 0 to 1 or from 1 to 0.

- Similarly in order to translate the object by one fourth i.e. by $2^{n-2}$ units, we have to replace the two most significant bits of the $y$ coordinate from 00 to 01, 01 to 10, 10 to 11 and 11 to 00.

- The Figure 18 gives the finite state transducers for these operations.

- For translation by an arbitrary power of 2, we can construct FSTs in a similar manner.

- To translate the object along the $x(z)$-axis, we have to change the corresponding $x(z)$ bits.

- To translate in the other direction (right to left) we have to subtract 1 instead of adding.
The FSTs for these operations can be easily constructed from the above FSTs.

In order to translate the object by any arbitrary value, say a units along the $x$ axis, b units along the $y$ axis and c units along the $z$ axis, we write $a(b,c)$ as a sum of powers of 2 and continuously apply the FSTs for each power of 2 along the $x(y,z)$ axis.

Figure 18: Translation of three dimensional object by $1/2$, $1/4$ of its size.
Transformations on Three Dimensional Objects (Contd ...)

- **Rotation of three dimensional objects**
  
  - For rotation of 3D objects we consider the coordinate axes to be shifted to the center of the three dimensional object so that the rotation would be in an anti clockwise direction about the center of the three dimensional object. Also the rotation is considered to be with respect to the z axis.
  
  - The rotation matrix for a rotation of angle $\theta$ about the z axis is given by
    
    $$
    \begin{pmatrix}
    \cos\theta & -\sin\theta & 0 \\
    \sin\theta & \cos\theta & 0 \\
    0 & 0 & 1 
    \end{pmatrix}
    $$
    
  - Here we consider the case of rotation through an angle of 45°. Let $x, y, z$ represent the address of the pixels of the three dimensional object in the given coordinate system.
Transformations on Three Dimensional Objects (Contd ...)

- If \( x', y', z' \) represent the address of the pixel in the new coordinate system, then the rotation through an angle of 45\(^\circ\) is specified as follows.

\[
x'' \leftarrow \frac{x' - y'}{\sqrt{2}}, \quad y'' \leftarrow \frac{x' + y'}{\sqrt{2}}, \quad z'' \leftarrow z'
\]

- As the division operation is not easily achieved using an FST we will consider the following transformation

\[
x'' \leftarrow \frac{x' - y'}{2}, \quad y'' \leftarrow \frac{x' + y'}{2}, \quad z'' \leftarrow z'
\]

- This transformation rotates the three dimensional object through an angle of 45\(^\circ\) and also scales the three dimensional object by a factor of \( \frac{1}{\sqrt{2}} \)

- Let \( M = (\Sigma, Q, \delta, I, F) \) be the FSA representing the three dimensional object. The operation of rotation of the three dimensional object by an angle of 45\(^\circ\) and scaling by a factor of \( \frac{1}{\sqrt{2}} \) is given by the following steps
Transformations on Three Dimensional Objects (Contd ...)

1. The FSA $M$ is scaled by a factor of $\frac{1}{2}$ to form the FSA $M'$. 

2. The transformation equations represented in terms of the old axes would be

$$x_1 \leftarrow x - (y - 2^{n-1}), \quad y_1 \leftarrow x + (y - 2^{n-1}), \quad z_1 \leftarrow z$$

where $(x_1, y_1, z_1)$ is obtained from $(x, y, z)$ after performing the following operations: scaling the given object, shifting the axes to the center of the three dimensional object, rotating the object by an angle of $45^\circ$ about the $z$ axis, and then shifting back to the old axes. So we first need to subtract $2^{n-1}$ from the $y$ coordinate. This can be achieved by changing the most significant bit of $y$ from 0 to 1 and 1 to 0. So we construct the FSA $M''$ from $M'$ as follows.

(a) $\Sigma'' = \Sigma', Q'' = Q', I'' = I', F'' = F'$

(b) $\delta''(q', 0) = \delta'(q', 2)$

(c) $\delta''(q', 1) = \delta'(q', 3)$

(d) $\delta''(q', 2) = \delta'(q', 0)$

(e) $\delta''(q', 3) = \delta'(q', 1)$

(f) $\delta''(q', 4) = \delta'(q', 6)$

(g) $\delta''(q', 5) = \delta'(q', 7)$

(h) $\delta''(q', 6) = \delta'(q', 4)$

(i) $\delta''(q', 7) = \delta'(q', 5)$

(j) $\delta''(q, a) = \delta'(q, a), \forall q \in Q', a \in \Sigma'$
3. We then apply the FST in Figure 19 to the FSA $M''$ to obtain the rotated FSA $N$. The FST in the Figure 19 performs a transformation such that the $x$ coordinate is replaced by $x - y$ and $y$ coordinate is replaced by $x + y$. The addition and subtraction are done bit by bit using a four state FST where each state represents a carry over from the previous calculation of 0 or 1 for each coordinate $x$ and $y$.

4. After shrinking the three dimensional object, we have applied the rotation transformation. The rotation transformation has a scale factor of $\sqrt{2}$ but the FST can only perform an one-one transformation. Hence there are pixels in the transformed object for which the values are not set. It is found that the transformation sets only pixels whose addresses end in 0 or 5 or 6. Hence in this step, we add transitions to the FSA $N$ so that the pixels with address ending in 0 or 5 or 6 also, if they are black make the pixels whose addresses end in 2 or 4 or 1 or 6 also black. Hence
Transformations on Three Dimensional Objects (Contd ...)

\[
\delta(q, 4) = \{ f \mid f \in F, f \in \delta(q, 0), \forall q \in Q_N \}
\]
\[
\delta(q, 1) = \{ f \mid f \in F, f \in \delta(q, 5), \forall q \in Q_N \}
\]
\[
\delta(q, 2) = \{ f \mid f \in F, f \in \delta(q, 0), \forall q \in Q_N \}
\]
\[
\delta(q, 4) = \{ f \mid f \in F, f \in \delta(q, 6), \forall q \in Q_N \}
\]

– The FSA N constructed at the end of the fourth step represents the a three dimensional object rotated by an angle of 45° and scaled by a factor of \( \frac{1}{\sqrt{2}} \)